

`rstudio::conf(2020L)`



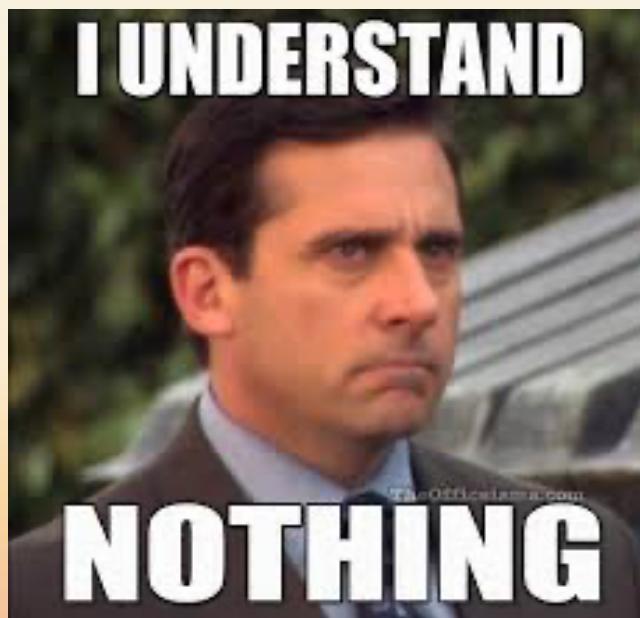
# List-columns in `data.table`



Tyson S. Barrett



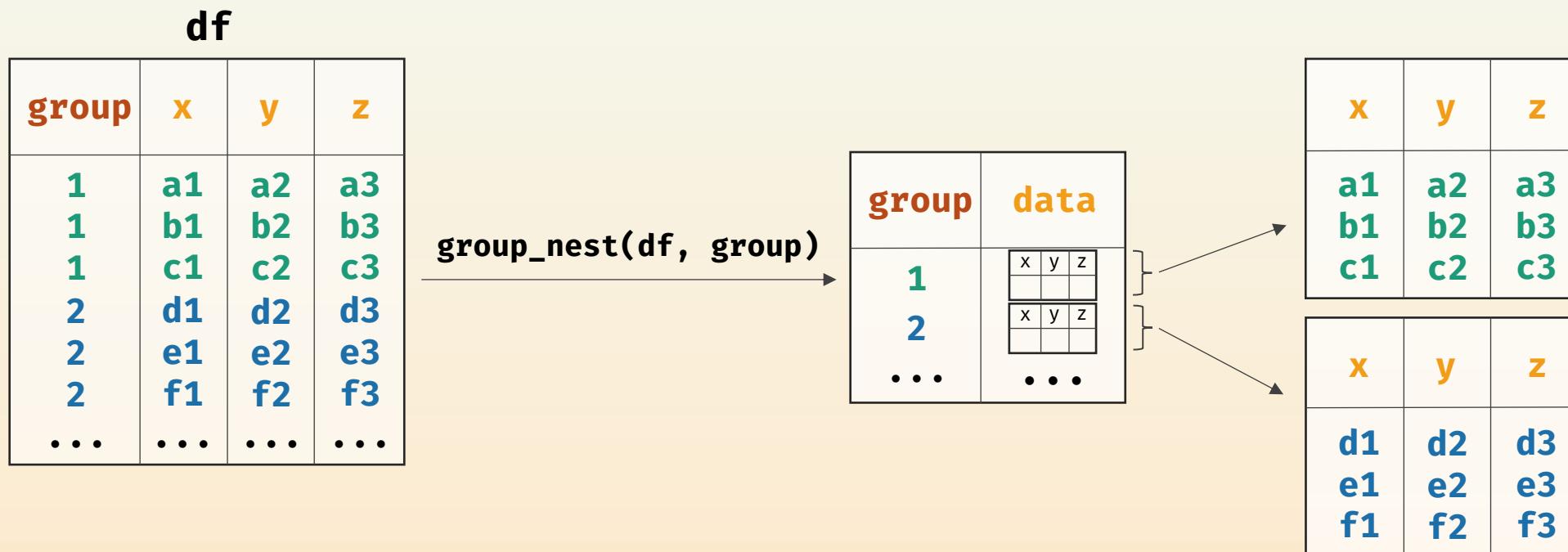
Artwork from [@juliesquid](#) for [@openscapes](#)  
(illustrated by [@allison\\_horst](#))

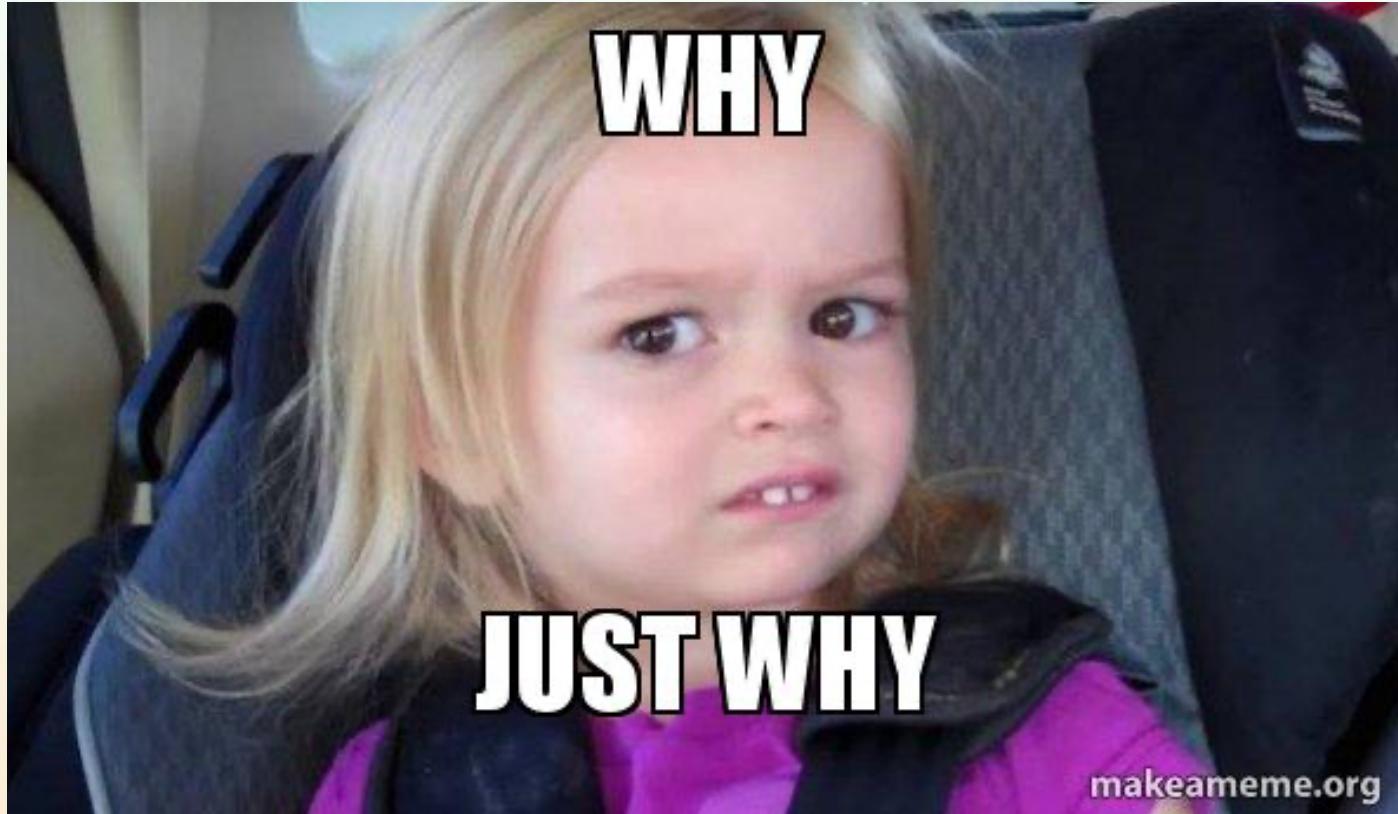


```
library(tidyverse)
df <- df %>%
  filter(x == 1) %>%
  mutate(z = y * 2)
```

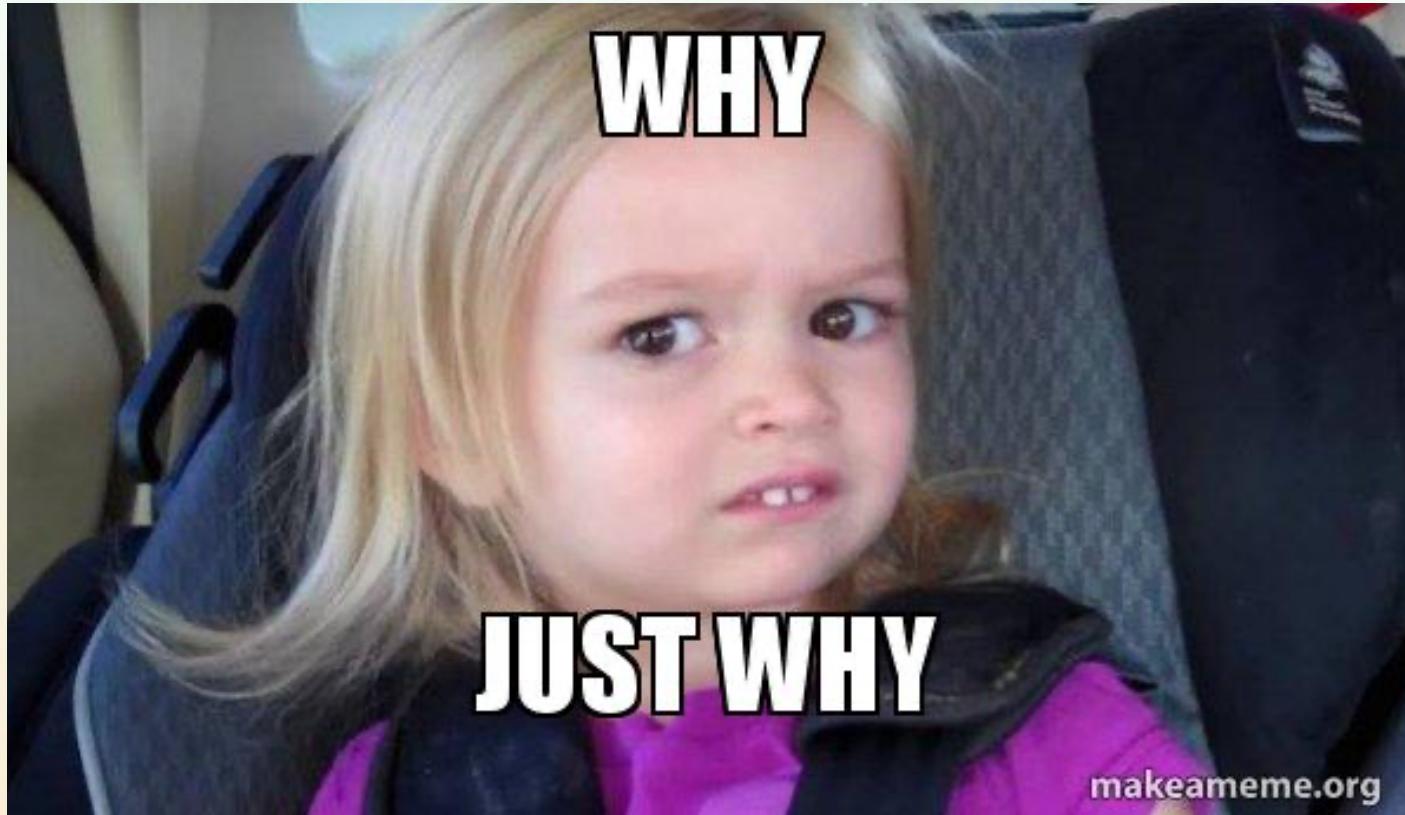
```
library(data.table)
dt[x == 1, z := y * 2]
```

# What are list-columns?





# Great question!



Cognitively easier to understand complex data

Show up all over the place  
(e.g. `bench::mark()`)

Has several functions in the **#tidyverse**

Can make data manipulations **safer**

Memory efficient  
**(fewer redundancies)**



# Why `data.table` ?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle management

Community

Feature rich



# Why `data.table` ?

**Concise syntax**

**Fast speed**

**Memory efficient**

Careful API lifecycle management

Feature rich



# Why `data.table` ?

Concise syntax

Fast speed

Memory efficient

Careful API interface

Feature rich

`dt[i, j, by]`

“filter”  
or  
“arrange”    “mutate”    “group\_by”

```
graph TD; dt[dt] --> i[i]; dt --> j[j]; dt --> by[by]; filter["filter"] --> i; arrange["arrange"] --> j; mutate["mutate"] --> by; group_by["group_by"] --> by;
```



# Why **data.table** ?

Concise syntax

Fast speed

Memory efficient

Careful API design

Feature rich

Responding to why **data.table** is so fast, Hadley Wickham:

“I think it’s a relentless focus on performance across the entire package.”

<https://twitter.com/hadleywickham/status/1153850194892640256>



# Why `data.table` ?

Concise syntax

Fast speed

Memory efficient

Careful API design

Feature rich

<https://h2oai.github.io/db-benchmark/>

## Summing by group

Input table: 1,000,000,000 rows x 9 columns ( 50 GB )

■	data.table	1.12.9	2019-12-12	1339s
■	(py)datatable	0.10.0a0	2019-11-24	2682s
■	dplyr	0.8.3	2019-12-05	timeout
■	pandas	0.25.3	2019-12-07	out of memory
■	spark	2.4.4	2019-12-05	not yet implemented
■	dask	2.9.0	2019-12-07	timeout
■	DataFrames.jl	0.20.0	2019-12-09	out of memory
■	ClickHouse	19.16.2.2	2019-12-09	CH server crash
■	cuDF	0.10.0	2019-12-05	out of memory
■	Modin		see README	pending

Let's  
Team  
Up





# The Grammar

**Nest** making a list-column of data tables or vectors

**Unnest** unnesting list-columns that have data tables or vectors in them



# A Brief Example!

Data from:

- [Statistica](#)
- [dplyr::starwars](#)

```
library(tidyverse)
library(data.table)
```

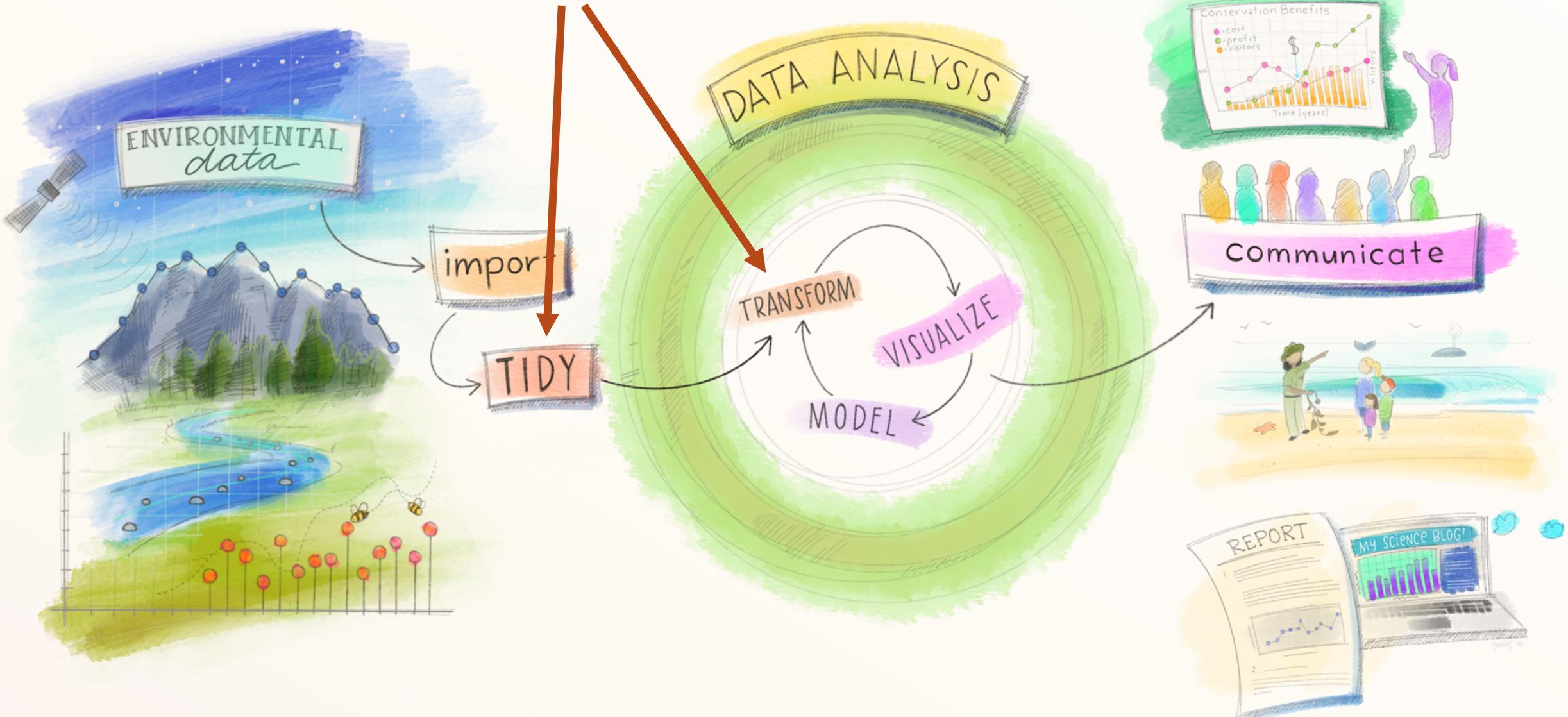
Note, the following approaches work with the current developmental version (1.12.9)

# The Data Sets

```
##      name                      species homeworld films ...
##      <chr>                     <chr>   <chr>     <list>
## 1 Luke Skywalker             Human    Tatooine  <chr [5]> ...
## 2 C-3PO                       Droid    Tatooine  <chr [6]> ...
## 3 R2-D2                       Droid    Naboo    <chr [7]> ...
## ...
```

```
##      films                    revenue
##      <char>                   <num>
## 1 A New Hope                  775.4
## 2 The Empire Strikes Back    538.4
## 3 Return of the Jedi          475.1
## 4 The Phantom Menace         1027.0
## ...
```

# You are here





# First, let's unnest

The variable `films` is currently a list-column of character vectors

```
sw <- as.data.table(dplyr::starwars)
sw <- sw[, .(films = films[[1]]),
         by = c(...)]
```



# First, let's unnest

The variable `films` is currently a list-column of character vectors

```
sw <- as.data.table(dplyr::starwars)  
sw <- sw[, .(films = films[[1]]),  
        by = c(...)]
```

The list-column

Other variables we want to  
keep in the data  
e.g. "name", "species"

# First, let's unnest

The variable `films` is currently a list-column of character vectors

```
sw <- as.data.table(dplyr::starwars)
sw <- sw[, .(films = films[[1]]),
         by = c(...)]
```

```
##           name          films ...
##      <char>        <char> ...
## 1: Luke Skywalker Revenge of the Sith ...
## 2: Luke Skywalker Return of the Jedi ...
## 3: Luke Skywalker The Empire Strikes Back ...
## ...
```



# Then, let's nest by film

Creates a variable called data with all the data associated with each film

```
swn <- sw[, .(data = list(.SD)),  
           by = films]
```

The data  
table

Name of new list  
column

Symbol for  
*Subset of  
the Data*

Tell it to make the  
list-column by each  
film



# The Nested Data

```
##      films          data
##      <char>        <list>
## 1: A New Hope <data.table>
## 2: Attack of the Clones <data.table>
## 3: Return of the Jedi <data.table>
## 4: Revenge of the Sith <data.table>
## 5: The Empire Strikes Back <data.table>
## 6: The Force Awakens <data.table>
## 7: The Phantom Menace <data.table>
```

# Joining Revenue with Nested

```
# join nested and revenue on films variable
swn <- swn[revenue, on = "films"]
swn[]
```

```
##      films          data      revenue
##      <char>        <list>      <num>
## 1: A New Hope <data.table> 775.4
## 2: The Empire  ... <data.table> 538.4
## 3: Return of   ... <data.table> 475.1
## 4: The Phantom ... <data.table> 1027.0
## ...
```

# Some analyses...

(Get counts of different species per film; Could use `dplyr` here instead)

```
# Get counts for each film from the data column  
swn[, counts := purrr::map(data, ~count(.x, species))]  
swn[]
```

Forces the  
print method

Creates a list column called  
“counts” of tibbles with count  
data by gender



# Some analyses...

(Get counts of different species per film; Could use `dplyr` here instead)

```
# Get counts for each film from the data column  
swn[, counts := purrr::map(data, ~count(.x, species))]  
swn[]
```

Note, no name because  
unnesting a data table

```
swun <- swn[, counts[[1]],  
            by = .(films, revenue)]  
swun[]
```

# The Unnested Data

```
##          films revenue species   n
##          <char>    <num>  <char> <int>
## 1: A New Hope  775.4   Droid     3
## 2: A New Hope  775.4 Human    12
## 3: A New Hope  775.4   Hutt      1
## 4: A New Hope  775.4 Rodian     1
## 5: A New Hope  775.4 Wookiee    1
## ...
```

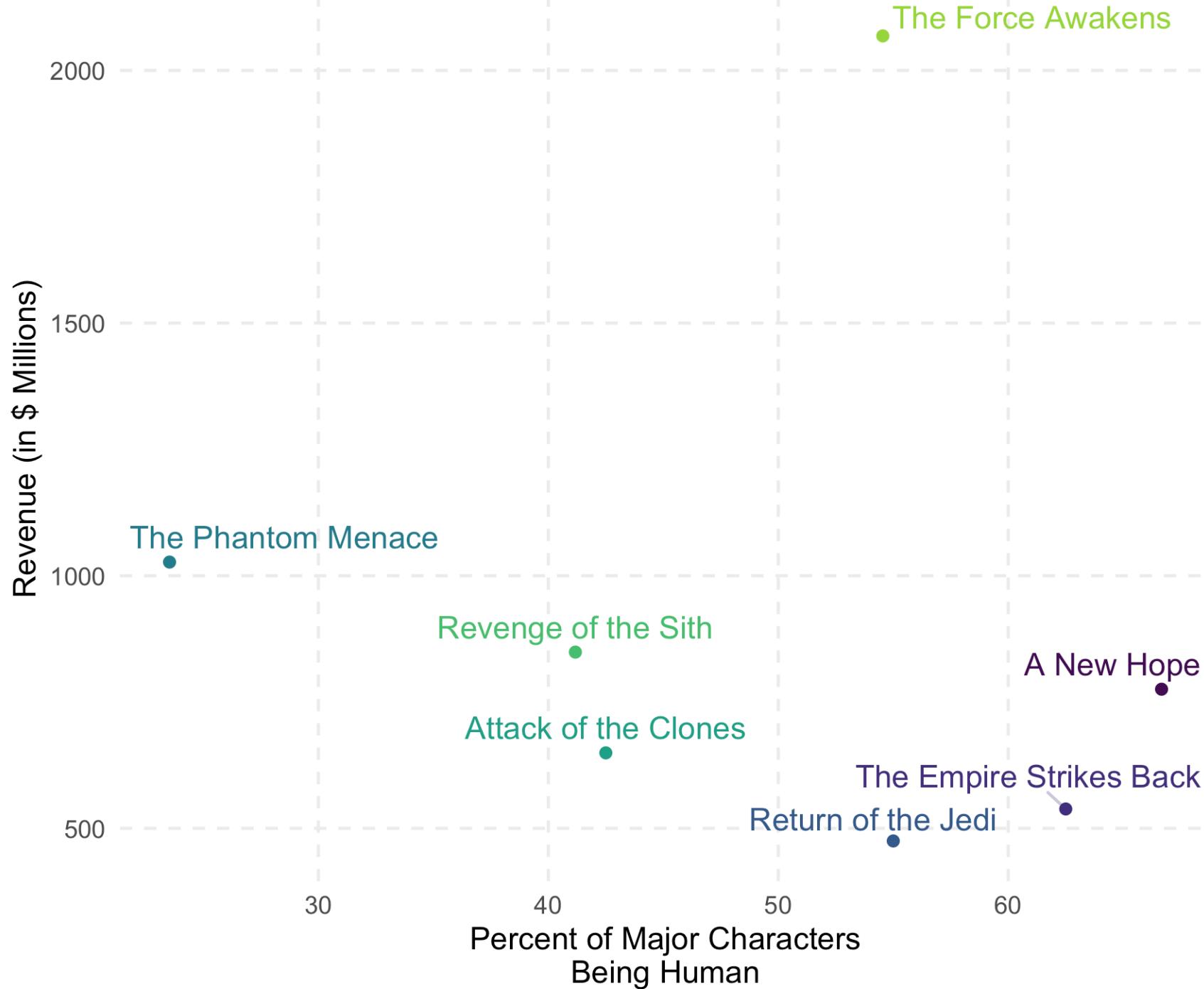
From here, we can get the proportion of females to all characters and check the relationship between proportion and revenue



# The

```
##  
##  
## 1 :  
## 2 :  
## 3 :  
## 4 :  
## 5 :  
## ..
```

From here  
the relation





# Cool, cool...

# But what about performance?!

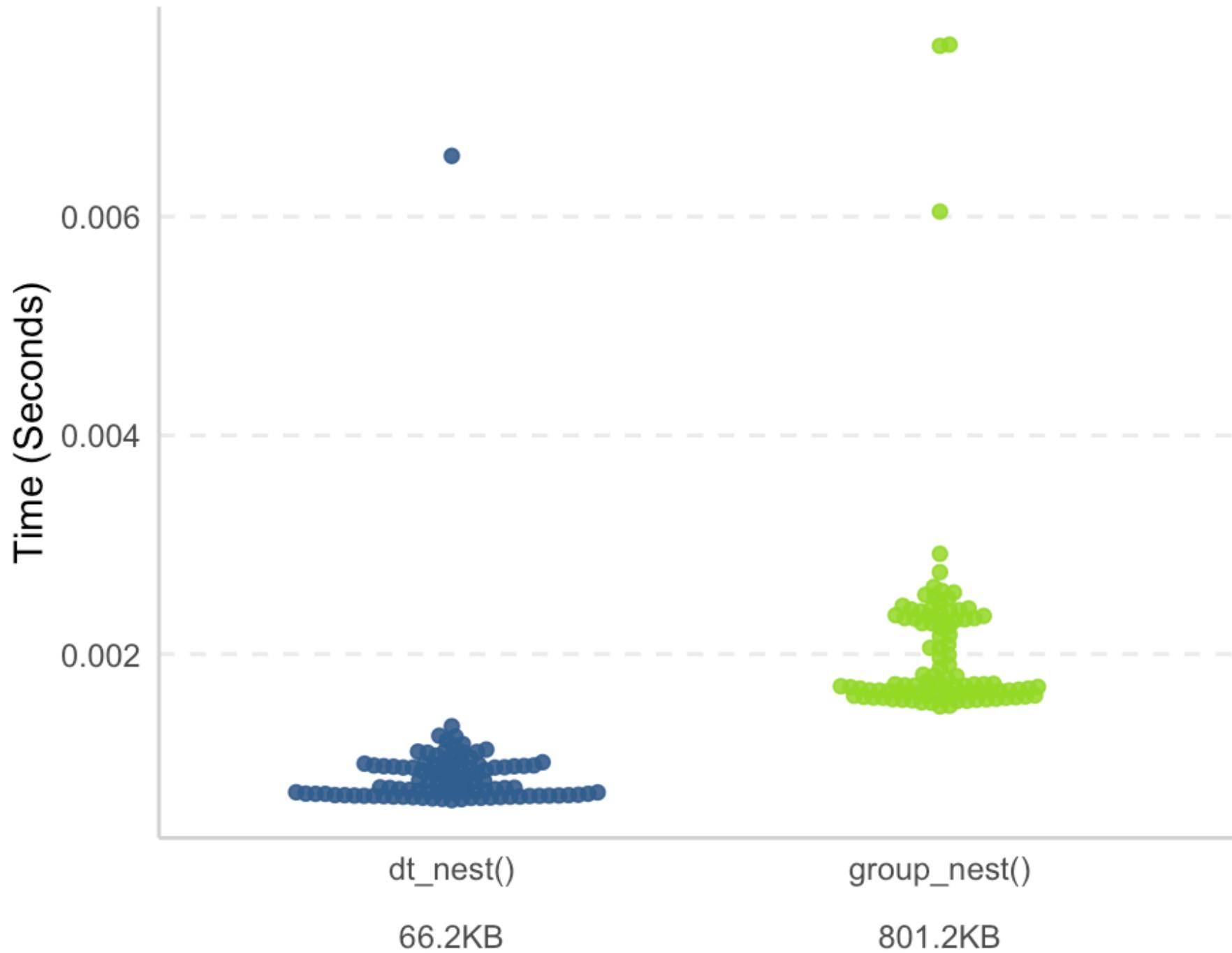
```
library(tidyfast)
```

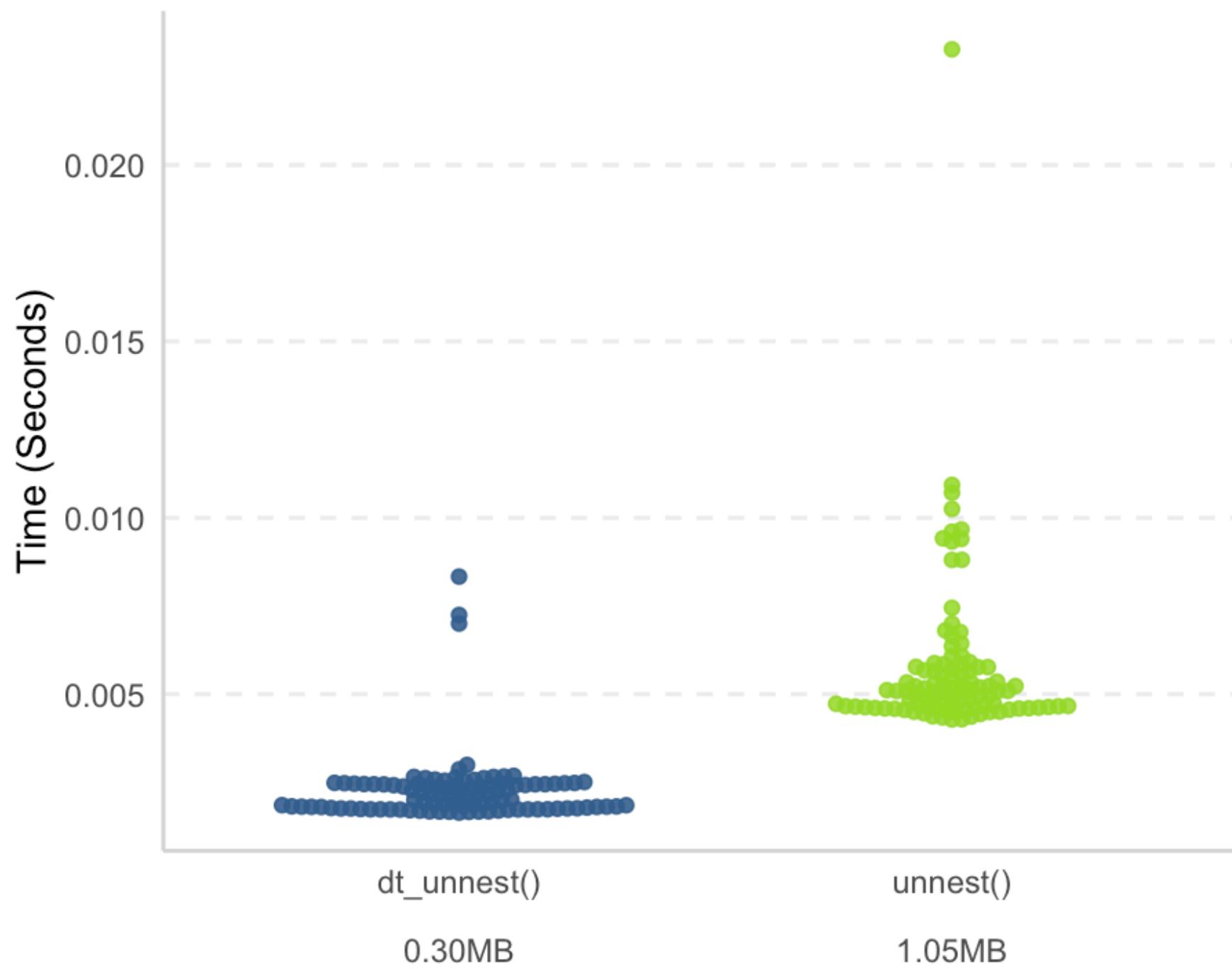
Packaged up the  
functions into  
**tidyfast**

```
remotes::install_github("tysonstanley/tidyfast")
```

Cool, cool  
But what

# Nesting





# Unnesting



# Tyson S. Barrett

Thanks to **Matt Dowle** and **Arun Srinivasan** and **the data.table team!**



[tyson.barrett@usu.edu](mailto:tyson.barrett@usu.edu)



[tysonbarrett.com](http://tysonbarrett.com)



[@healthandstats](https://twitter.com/healthandstats)



[github.com/tysonstanley](https://github.com/tysonstanley)



Slides at [tysonbarrett.com/teaching](http://tysonbarrett.com/teaching)